

# Software-Projekt I

Prof. Dr. Rainer Koschke

Arbeitsgruppe Softwaretechnik  
Fachbereich Mathematik und Informatik  
Universität Bremen

Sommersemester 2013

## Vorbemerkungen I

### Vorbemerkungen

Ziele und Inhalt

Übungsbetrieb

Angestrebte Resultate

Aktivitäten der Software-Entwicklung

Projektablauf

Scheinbedingungen

Anmeldung

Aufgabe im SWP II

Ressourcen

Lehrbücher

# Anmerkungen zu diesem Skriptum

Einige dieser Folien basieren auf den Folien oder Skripten von...

- Prof. Dr. Jochen Ludewig (2003), Universität Stuttgart
- Prof. Dr. Susanne Maaß (2001), Universität Bremen
- Prof. Dr. Karl-Heinz Rödiger (2004), Universität Bremen
- Prof. Dr. Andreas Winter (2003), Universität Oldenburg

Ich danke für deren freundliche Genehmigung.

3 / 24

## Die zehn teuflischen SWP-Gebote

- 1 Du sollst nicht in die Vorlesung gehen.
- 2 Wenn (1) doch geschieht, dann sollst du nur spielen und surfen.
- 3 Du sollst die Vorlesung nicht vor- und nachbereiten und keine Notizen machen.
- 4 Du sollst keinesfalls Fragen während der Vorlesung stellen.
- 5 Du sollst nicht in die Übungen gehen.
- 6 Du sollst niemals die Übungszettel vor der Prüfung anschauen.
- 7 Du sollst die Musterklausur erst nach der Klausur anschauen.
- 8 Wenn (7) doch geschieht, sollst du keinesfalls die Musterklausur vor der Klausur lösen.
- 9 Du sollst frühestens drei Tage vor der Prüfung anfangen zu lernen.
- 10 Du sollst nach dem Nichtbestehen nicht in die Klausureinsicht gehen.

6 / 24

# Ziele der Vorlesung

Primäre Ziele:

- Rüstzeug für erfolgreiche Durchführung eures Software-Projekts vermitteln
  - Modell für zukünftige ähnliche Projekte bieten
- Praktikum und Vorlesung bilden eine Einheit

Primäre Ziele sind **nicht**:

- Vollständige Darstellung aller Themengebiete der Softwaretechnik<sup>1</sup>
- Vermittlung von spezifischen Kenntnissen für die Entwicklung des Anwendungssystems X

---

<sup>1</sup>Wird im Hauptstudium nachgereicht.

## Szenario dieses Software-Projekts

- Erstellung eines großen Softwaresystems
  - mehrere Mitarbeiter über einen langen Zeitraum
  - **nicht**: ein „Freizeit“-Programmierer
- für einen realen Auftraggebers
  - Individualsoftware
  - **nicht**: Standardsoftware
- im Rahmen eines Projekts
  - einmalige Zielverfolgung
  - dennoch: Software soll wartbar sein
  - **nicht**: Weiterentwicklung existierender Software  
(aber: die Software wird von uns später weiterentwickelt;  
Wartbarkeit ist erklärtes Ziel)

# Stufen zum Erfolg

- zu votierende Übungsaufgaben mit Tutorium
- in Vorlesung und Tutorium integrierter Technikkurs
- Tools, Frameworks und Beispiele im Tutorium
- Mini-Projekt begleitend zu SWP-I:
  - Abgabe von Dokumenten während der Vorlesungszeit
  - Implementierung und Test im einwöchigen Blockkurs in vorlesungsfreier Zeit (Durchstich für 2-3 Anwendungsfälle)
- im WS: SWP-II-Projekt: Ausbau der Ergebnisse aus SWP-I

9 / 24

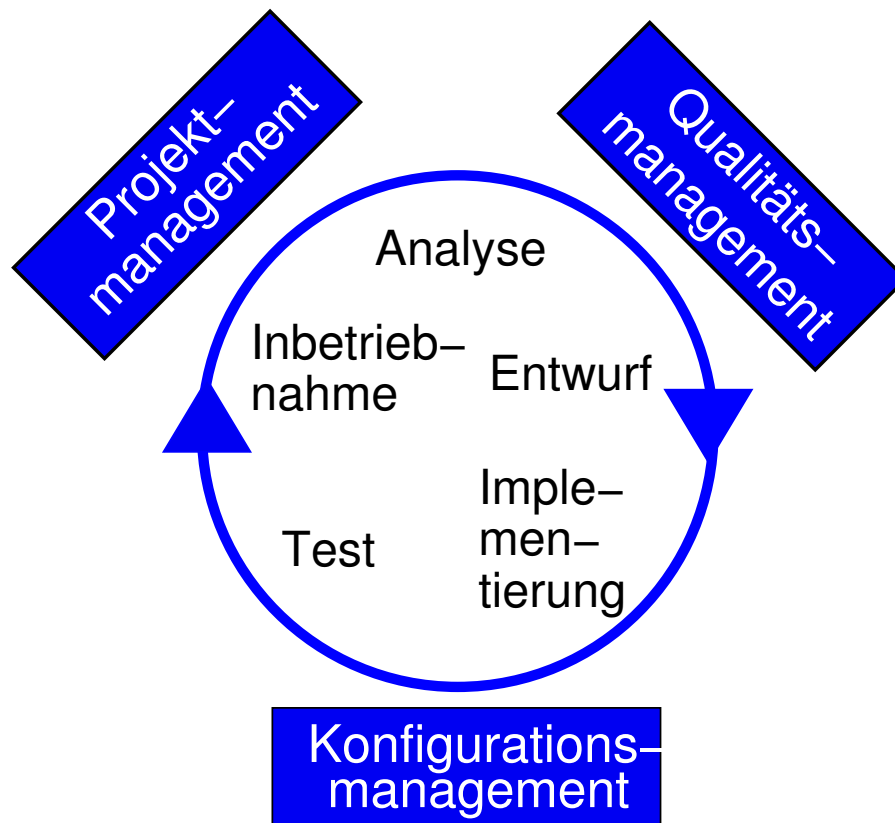
## Angestrebte Resultate

- Softwaretechnik ist nicht nur Programmieren.
- Softwaretechnik ist auch Programmieren.
- Software-Entwicklung produziert Dokumente.
- Der Quellcode ist **ein** Dokument **unter vielen**.

Ihr sollt nicht nur ein richtiges System bauen.  
Ihr sollt ein System auch richtig bauen.

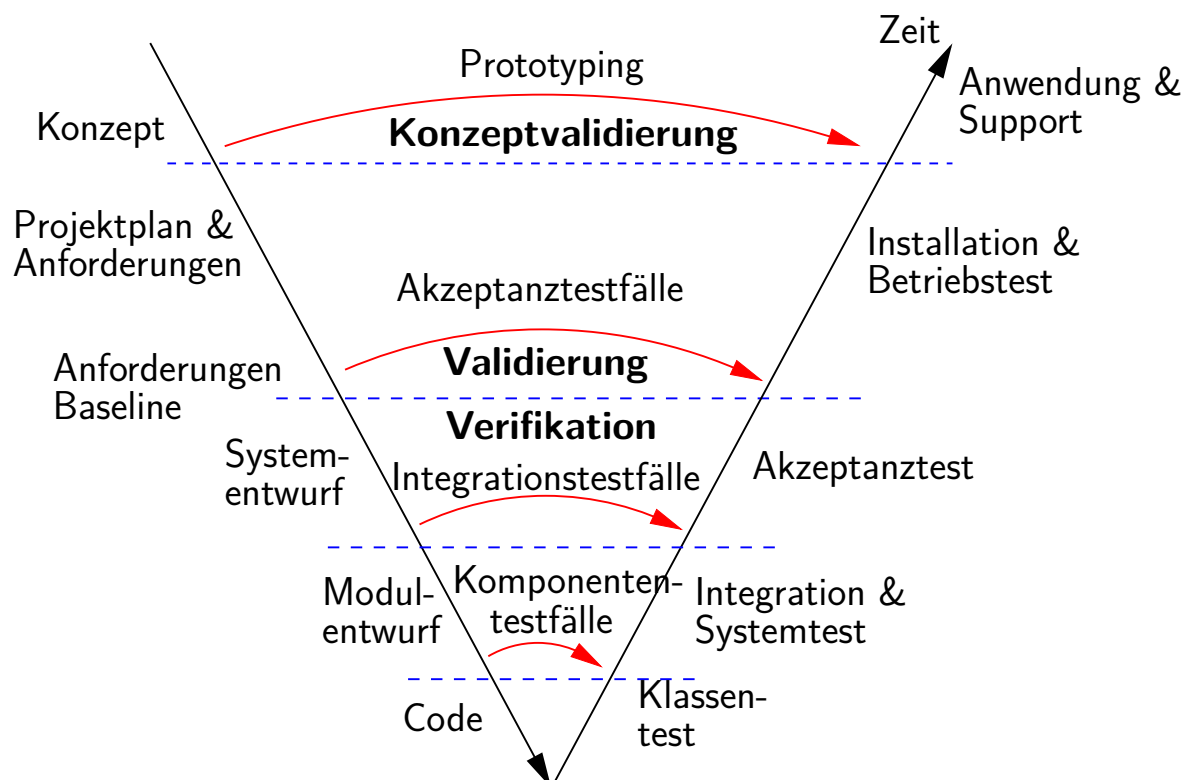
10 / 24

# Aktivitäten bei der Softwareentwicklung



11 / 24

## V-Modell von Boehm (1979)

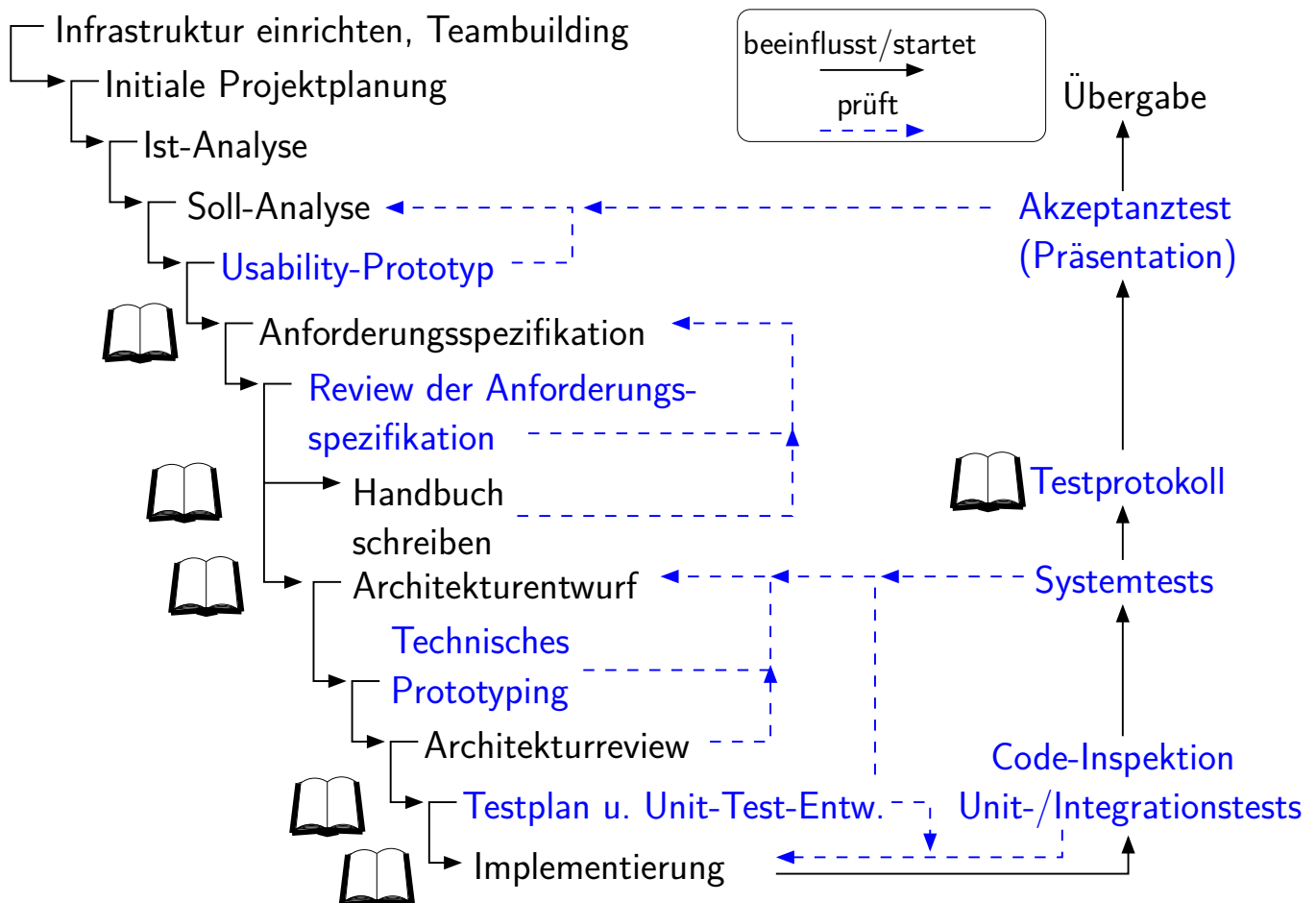


12 / 24

Das V-Modell ist kein eigenständiges Prozessmodell im eigentlichen Sinn. Die Produkte im V-Modell werden wie im Wasserfallmodell streng sequenziell erstellt. Es fügt dem Wasserfallmodell lediglich eine zusätzliche strukturelle Sicht hinzu, nämlich dass für jedes zu erstellende Dokument ein entsprechender Test existieren und durchgeführt werden soll.

Das V-Modell sieht sowohl Verifikationen als auch Validierungen vor. Validierung: Es wird das richtige Produkt erstellt. Verifikation: Das Produkt wird richtig erstellt.

Eine weitere Konsequenz des V-Modells für die konkrete Projektplanung ist, dass man die Test- und Validierungsaktivitäten früher als die tatsächliche Durchführung der Aktivität vorbereiten kann. Beispielsweise kann der Akzeptanztest vorbereitet werden, sobald man die Anforderungen kennt. Auf diese Weise können Aktivitäten bei einem Projektteam parallelisiert werden: Der Tester kann Testfälle für den Akzeptanztest entwickeln, auch wenn noch keine Implementierung existiert. Dies hat nicht nur den Vorteil, dass Tests nicht einfach dem Zeitdruck zum Opfer fallen können, sondern auch, dass alle Dokumente schon sehr früh einer Prüfung unterzogen werden: Nur wenn ein Dokument präzise, widerspruchsfrei und vollständig ist, kann ein Testfall dafür angegeben werden. Tester können also schon frühzeitig Mängel eines Dokuments aufzeigen. Und es gilt: Je früher ein Mangel erkannt wird, desto billiger ist seine Beseitigung.



## Scheinbedingungen

`http://www.informatik.uni-bremen.de/st/Lehre/swpI\_13/scheinbedingungen.html`

Ausblick Wintersemester 2013/14:

- Teilnahme auch ohne SWP-I-Schein möglich
- **aber:** keine gemischten Gruppen von Teilnehmern mit und ohne SWP-I-Schein

Anmeldung bis zum bis 4.04., 23.59 Uhr<sup>2</sup> über MEMS:

http:

`//www.informatik.uni-bremen.de/st/mems/courses.php?lng=de`

- nur in Sechsergruppen anmelden
- nur mit TZI-Email-Adresse anmelden
- Gruppennamen sorgfältig auswählen

---

<sup>2</sup>Ortszeit Bremen

## Technische Randbedingungen

- Implementierung in Java 5 oder höher
- Mehrbenutzerbetrieb
- Client/Server-Anwendung
- relationale Datenbank muss für die Persistenz benutzt werden:

Persistenz-Frameworks erlaubt, aber dann deklarative Verwendung von SQL-ähnlichen Abfragen verlangt

- Plattformen: Windows, Linux, Android







- Web-Seite zur Vorlesung:  
[http://www.informatik.uni-bremen.de/st/Lehre/swpI\\_13/](http://www.informatik.uni-bremen.de/st/Lehre/swpI_13/)
  - Folien mit Kommentaren und Folien mit Annotationen aus der Vorlesung bei Stud.IP:  
<http://elearning.uni-bremen.de/>
- dort registrieren
- Video-Aufzeichnung aus dem letzten Jahr:  
<http://mlecture.uni-bremen.de>

19 / 24





## Lehrbücher zur Softwaretechnik I

Allgemeine Literatur zur Softwaretechnik:


-  Balzert (1998): Umfassendes deutsches Lehrbuch vmtl. der Bestseller in Deutschland; leider nicht mehr im Buchhandel verfügbar.
-  Balzert (2009): Neueste Auflage des Lehrbuchs von Balzert, enthält jedoch nur Basiskonzepte und Anforderungsanalyse
-  Balzert (2008): Neueste Auflage des Lehrbuchs von Balzert zum Thema Software-Management (Planung, Risiken, Vorgehensmodelle).
-  Ludewig und Lichter (2006): Umfassendes Lehrbuch, das aus Ludewigs Vorlesungen rund um Softwaretechnik entstanden ist. Diese Vorlesung basiert in großen Teilen auf dem Skript von Ludewigs Vorlesung.

20 / 24

## Lehrbücher zur Softwaretechnik II

-  Sommerville (2012): Ein Standardlehrbuch, sowohl in deutscher als auch englischer Sprache verfügbar. Im Umfang vergleichbar mit dem Buch von Pressman (2003).
-  Pressman (2003): Ein umfassendes englisches Lehrbuch, das man fast schon als Enzyklopädie bezeichnen könnte. Behandelt auch nicht-objektorientierte Konzepte.
-  Brügge und Dutoit (2004): Eine Einführung in Deutsch mit Schwerpunkt Objektorientierung und UML.
-  Zuser u. a. (2004): Eine Einführung in Deutsch mit Schwerpunkt Objektorientierung und dem Rational Unified Process. Weniger umfassend als das Buch von Brügge und Dutoit (2004).

Spezialthemen:

-  Störrle (2005): Eine kurze Einführung in die Konzepte der UML 2.0 in deutscher Sprache.

21 / 24

- 1 Balzert 1998** BALZERT, Helmut: Lehrbuch der Software-Technik. Spektrum Akademischer Verlag, 1998. – derzeit nicht mehr verfügbar, wird neu aufgelegt. – ISBN 978-3827403018
- 2 Balzert 2008** BALZERT, Helmut: Lehrbuch der Softwaretechnik: Softwaremanagement. 2. Aufl. Spektrum Akademischer Verlag, 2008. – ISBN 978-3-8274-1161-7
- 3 Balzert 2009** BALZERT, Helmut: Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering. 3. Aufl. Spektrum Akademischer Verlag, 2009. – ISBN 978-3-8274-1705-3
- 4 Boehm 1979** BOEHM, Barry W.: Guidelines for verifying and validating software requirements and design specifications. In: EURO IFIP 79, IFIP, 1979, S. 711–719
- 5 Brügge und Dutoit 2004** BRÜGGE, Bernd ; DUTOIT, Allen H.: Objektorientierte Softwaretechnik. Prentice Hall, 2004

22 / 24

- 6 Ludewig 2003** LUDEWIG, Jochen: Einführung in die Softwaretechnik. Vorlesungs-Skriptum. 2003
- 7 Ludewig und Lichter 2006** LUDEWIG, Jochen ; LICHTER, Horst: Software Engineering – Grundlagen, Menschen, Prozesse, Techniken. dpunkt.verlag, 2006
- 8 Maaß 2001** MAASS, Susanne: Soziotechnische Systeme. Vorlesungs-Skriptum. 2001
- 9 Pressman 2003** PRESSMAN, Roger: Software Engineering – A Practitioner's Approach. Fünfte Ausgabe. McGraw-Hill, 2003
- 10 Rödiger 2004** RÖDIGER, Karl-Heinz: Vortrag Rechtlicher Rahmen der Software-Entwicklung in der Vorlesung Software-Projekt. Vorlesungs-Skriptum. 2004
- 11 Sommerville 2012** SOMMERVILLE, Ian: Software Engineering. Auflage 1. Pearson Studium, 2012. – ISBN 978-3868940992
- 12 Störrle 2005** STÖRRLE, Harald: UML 2 für Studenten. Pearson Studium, 2005. – ISBN 3-8273-7143-0

23 / 24

- 13 Winter 2003** WINTER, Andreas: Einführung in die Softwaretechnik. Vorlesungs-Skriptum. 2003
- 14 Zuser u. a. 2004** ZUSER, W. ; GRECHENIG, T. ; KÖHLE, M.: Software Engineering mit UML und dem Unified Process. Zweite Ausgabe. Pearson Studium, 2004

24 / 24